Practical Verification of Neural Network Enabled State Estimation System for Robotics

Wei Huang[1], Yifan Zhou[1], Youcheng Sun[2], James Sharp[3], Simon Maskell[1], and Xiaowei Huang[1]

***Abstract*— We study for the first time the verification problem on learning-enabled state estimation systems for robotics, which use Bayes filter for localisation, and use deep neural network to process sensory input into observations for the Bayes filter. Specifically, we are interested in a robustness property of the systems: given a certain ability to an adversary for it to attack the neural network without being noticed, whether or not the state estimation system is able to function with only minor loss of localisation precision? For verification purposes, we reduce the state estimation systems to a novel class of labelled transition systems with payoffs and partial order relations, and formally express the robustness property as a constrained optimisation objective. Based on this, practical verification algorithms are developed. As a major case study, we work with a real-world dynamic tracking system that uses a Kalman filter (a special case of the Bayes filter) to localise and track a ground vehicle. Its perception system, based on convolutional neural networks, processes a high-resolution Wide Area Motion Imagery (WAMI) data stream. Experimental results show that our algorithms can not only verify the robustness of the WAMI tracking system but also provide useful counterexamples.**

I. Introduction

State estimation systems have been widely deployed for many tasks within robotic applications, including localisation [1], tracking [2], and control [3]. This paper considers neural network enabled state estimation systems where neural networks are used to process perceptional input received via sensors. More and more robotics applications adopt neural network components to take advantage of their high prediction precision [4].

However, the neural network has been found to be vulnerable to adversarial attacks, i.e., a small, imperceptible perturbation on the input may alter the classification output. The concern has been raised on how safe a learning enabled system is when learning components interact with other components (including Bayes filter components). In [5], it has been found that the system is able to compensate (to some degree) against adversarial attacks on its neural network component, but there may also be new uncertainties from the interactions between learning and non-learning components. While *some* vulnerability cases were reported in [5], there is no comprehensive study on *all* potential risks in a learning enabled system, from the perspective of formal verification. Formal verification can prove that a system is correct against *all* possible risks over a specification and the formal model of the system, and returns counterexamples when it cannot. The ability to sufficiently identify risks is necessary for the deployment of safety critical applications – this paper addresses this need. It is the *first time a verification approach has been developed for securing learning enabled state estimation systems*.

Technically, we first formalise a state estimation system as a novel labelled transition system which has components for payoffs and partial order relations. Specifically, every transition is attached with a payoff, and for every state there is a partial order relation between its out-going transitions from the same state. Second, we show that the verification of the robustness property on such a system can be reduced into a constrained optimisation problem. Third, to enable practical verification, we develop two algorithms: (1) a verification algorithm – that can achieve complete results but cannot be used for run-time – and (2) a heuristic algorithm – that can be used efficiently in run-time, perform well in most cases, but cannot provide a completeness guarantee.

As a major case study, we work with a real-world dynamic tracking system [3], which detects and tracks ground vehicles over the high-resolution Wide Area Motion Imagery (WAMI) data stream, named *WAMI tracking system* in this paper. We apply the developed algorithms to the WAMI tracking system for safety analysis, in particular, we consider on a robustness property that concerns whether the system can function well under attack on the perceptional neural network components.

II. Preliminaries

A. Neural Networks

Let $\mathtt{X}$ be the input domain and $\mathtt{Y}$ be the set of labels. A neural network $N : \mathtt{X} \to \mathcal{D}(\mathtt{Y})$ can be seen as a function mapping from $\mathtt{X}$ to probabilistic distributions over $\mathtt{Y}$. That is, $N(x)$ is a probabilistic distribution, which assigns for each label $y \in \mathtt{Y}$ a probability value $(N(x))_y$. We let $f_N : \mathtt{X} \to \mathtt{Y}$ be a function such that for any $x \in \mathtt{X}$, $f_N(x) = \arg\max_{y \in \mathtt{Y}}\{(N(x))_y\}$, i.e., $f_N(x)$ returns the classification.

B. Neural Network Enabled State Estimation

We consider a time-series linear state estimation problem that is widely assumed in the context of object tracking. The process model is defined as follow.

$$\mathbf{s}_k = \mathbf{F} \cdot \mathbf{s}_{k-1} + \omega_k \tag{1}$$

where $\mathbf{s}_k$ is the state at time k, $\mathbf{F}$ is the transition matrix, ω_k is a zero-mean Gaussian noise such that $\omega_k \sim \mathcal{N}(0, \mathbf{Q})$, with $\mathbf{Q}$ being the covariance of the process noise. Usually, the

[1]Wei Huang, Yifan Zhou, Simon Maskell, and Xiaowei Huang are with the school of Electrical Engineering, Electronics and Computer Science (EEECS), University of Liverpool, UK.

[2]Youcheng Sun is with School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK.

[3]James Sharp is with Defence Science and Technology Laboratory, UK.

states are not observable and need to be determined indirectly by measurement and reasoning. The measurement model is:

$$\mathbf{z}_k = \mathbf{H} \cdot \mathbf{s}_k + \mathbf{v}_k \tag{2}$$

where $\mathbf{z}_k$ is the observation, $\mathbf{H}$ is the measurement matrix, $\mathbf{v}_k$ is a zero-mean Gaussian noise such that $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$, and $\mathbf{R}$ is the covariance of the measurement noise.

Bayes filters have been used for reasoning about the observations, $\{\mathbf{z}_k\}$, with the goal of learning the underlying states $\{\mathbf{s}_k\}$. A Bayes filter maintains a pair of variables, $(\mathbf{s}_k, \mathbf{P}_k)$, over the time, denoting Gaussian estimate and uncertainty, respectively. The basic procedure of a Bayes filter is to use a transition matrix, $\mathbf{F}_k$, to predict the current state, $(\hat{\mathbf{s}}_k, \hat{\mathbf{P}}_k)$, given the previous state, $(\mathbf{s}_{k-1}, \mathbf{P}_{k-1})$. The prediction state can be updated into $(\mathbf{s}_k, \mathbf{P}_k)$ if a new observation, $\mathbf{z}_k$, is obtained. In the context of the aforementioned problem, this procedure is iterated for a number of time steps, and is always discrete-time, linear, but subject to noises.

We take the Kalman Filter (KF), one of the most widely used variants of Bayes filter, as an example to demonstrate the above procedure. Let $\mathbf{s}_0 \in \mathbb{R}^n \sim \mathcal{N}(\hat{\mathbf{s}}_0, \hat{\mathbf{P}}_0)$ be the initial state, such that $\hat{\mathbf{s}}_0 \in \mathbb{R}^n$ and $\hat{\mathbf{P}}_0 \in \mathbb{R}^{n \times n}$ represent our knowledge about the initial estimate and the corresponding covariance matrix, respectively.

First, we perform the **state prediction** for $k \geq 1$:

$$\begin{array}{rcl} \hat{\mathbf{s}}_k & = & \mathbf{F}_k \mathbf{s}_{k-1} \\ \hat{\mathbf{P}}_k & = & \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_k \end{array} \tag{3}$$

Then, we can **update the filter**:

$$\begin{array}{rcl} \mathbf{s}_k & = & \hat{\mathbf{s}}_k + \mathbf{K}_k \mathbf{y}_k \\ \mathbf{P}_k & = & (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \hat{\mathbf{P}}_k \end{array} \tag{4}$$

such that

$$\begin{array}{rcl} \mathbf{y}_k & = & \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{s}}_k \\ \mathbf{S}_k & = & \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \\ \mathbf{K}_k & = & \hat{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1} \end{array} \tag{5}$$

Intuitively, $\mathbf{y}_k$ is usually called "innovation" in signal processing that represents the difference between the real observation and the predicted observation, $\mathbf{S}_k$ is the covariance matrix of this innovation, and $\mathbf{K}_k$ is the Kalman gain, representing the relative importance of innovation $\mathbf{y}_k$ with respect to the predicted estimate $\hat{\mathbf{s}}_k$.

In a neural network enabled state estimation, a perception system – which may include multiple CNNs – will provide a set of candidate observations Z_k, any of which can be chosen as the new observation $\mathbf{z}_k$. From the perspective of robotics, Z_k includes a set of possible states of the robot, measured by (possibly several different) sensors at time k. These measurements are imprecise, and are subject to noises from both the environment (called epistemic uncertainty) and the imprecision of sensors (aleatory uncertainty).

C. A Real-World WAMI Dynamic Tracking System

In this part, we have a brief introduction to the real-world WAMI dynamic tracking system that will be used as our major case study. The details of this system can be found in [5] and [3]. The tracking system requires continuous imagery input from e.g., airborne high-resolution cameras. The input is a video, which consists of a finite sequence of WAMI images. Each image contains a number of vehicles. The processing chain of the WAMI tracking system is as follows.

1) Align a set of previous frames with the incoming one.
2) Construct the background model of incoming frames using the median frame.
3) Extract moving objects using background subtraction.
4) Determine if the moving objects are vehicles by using a Binary convolutional neural networks (CNN).
5) For complex cases, predict the locations of moving objects/vehicles using a regression CNN.
6) Track one of the vehicles using a Kalman filter.

WAMI tracking uses **Gated Nearest Neighbour (Gnn)** to choose the new observation $\mathbf{z}_k$: from the set Z_k, the one closest to the predicted measurement $\mathbf{H}_k \cdot \hat{\mathbf{s}}_k$ is chosen, i.e.,

$$\mathbf{z}_k = \arg\min_{\mathbf{z} \in Z_k} ||\mathbf{z} - \mathbf{H}_k \cdot \hat{\mathbf{s}}_k||_p \tag{6}$$

$$s.t. \quad ||\mathbf{z} - \mathbf{H}_k \cdot \hat{\mathbf{s}}_k||_p \leq \epsilon_k \tag{7}$$

where $||\cdot||_p$ is for the L^p-norm distance (p=2, i.e., Euclidean distance, is used in this paper), and ϵ_k is the gate value, representing the maximum uncertainty the system is able to work with.

Specifically, the WAMI system has the following $\mathbf{s}$ and $\mathbf{P}$:

$$\mathbf{s} = \begin{bmatrix} l \\ v \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \Sigma_{ll} & \Sigma_{lv} \\ \Sigma_{vl} & \Sigma_{vv} \end{bmatrix} \tag{8}$$

where $\mathbf{s}$ contains the currently best estimates – or mean values – of two variables l, representing the location, and v, representing the velocity, respectively. Elements of $\mathbf{P}$, Σ_{ij}, represent the degrees of correlation between variables $i, j \in \{l, v\}$. The diagonal of $\mathbf{P}$ contains the mean square error of the estimate $\mathbf{s}$. The **uncertainty – or Bayesian uncertainty –** ϵ is the trace of the covariance matrix:

$$\epsilon = tr(\mathbf{P}) = \Sigma_{ll} + \Sigma_{vv} \tag{9}$$

Intuitively, ϵ denotes a search range and only within this range, observations are considered. Normally, ϵ will gradually shrink before being bounded – a convergence property of the KF as explained below.

We remark that this WAMI dynamic tracking system models a Poisson-Bernoulli mixture process. The track initialisation and the occurrence of false alarms follow a Poisson distribution (since we are focusing on one single target, we choose not to explicitly model this part) and the detectability of the target follows a Bernoulli distribution which models the CNN measurements (including mis-detections and spatial errors). In this paper, KF is used to address the measurement noise, but we note that it cannot be ensured that the Gaussian noise is sufficient in this case. Therefore, the usage of KF here is only under the assumption of Gaussian noises whose parameters are empirically configured. It can be seen as a smoothing algorithm rather than an optimal solution. Nevertheless, investigating alternative likelihoods that more faithfully represent the uncertainty and automated parameter estimation for this problem is an intriguing future work.

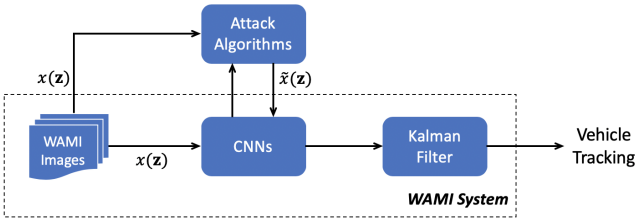


Fig. 1: The workflow of attacking the WAMI system.

D. Expansion of Bayesian Uncertainty in Kalman Filters

Generally, a KF system works with the two phases, prediction (3) and update (4), alternating. Theoretically, KFs converge [6] under a good set of parameters $\mathbf{F}$, $\mathbf{H}$, $\mathbf{Q}$, and $\mathbf{R}$. In this paper, we assume that the KF system has been well designed to ensure the convergence. Empirically, this has been proven possible in many practical systems. We are interested in another property of KF, i.e., the uncertainty in $\hat{\mathbf{P}}_k$ increases as opposed to $\mathbf{P}_k$; if no update phase, i.e., Equation (4), is held, this predicted covariance $\hat{\mathbf{P}}_k$ will be carried over to the next step as $\mathbf{P}_{k+1}$. In the WAMI tracking system, when observation $\mathbf{z}_k$ is unavailable within ϵ for some reason, the update step can be skipped and multiple prediction steps are performed consecutively. In this case, the Bayesian uncertainty ϵ may 'explode', and the search range of observations is expanded. We will explain later this property can be utilised to design a monitor to counter the attack, and therefore should be considered when analysing the robustness.

III. Problem Formulation

A. Threat Model of Adversarial Attack on Perception System

In Section II-C, a neural network based perception system determines whether or not there is a vehicle at a location $\mathbf{z}$. Let $x(\mathbf{z}) \in \mathbb{R}^{d_1 \times d_2}$ be an image covering the location $\mathbf{z}$, a neural network function $f_N : \mathbb{R}^{d_1 \times d_2} \rightarrow \{0, 1\}$ maps $x(\mathbf{z})$ into a Boolean value $f_N(x(\mathbf{z}))$ representing whether or not a vehicle is present at location $\mathbf{z}$. There are two types of erroneous detection: (1) a wrong classification prediction of the image $x(\mathbf{z})$, and (2) a wrong positioning of a moving object within $x(\mathbf{z})$. We focus on the former since the WAMI tracking system has a comprehensive mechanism to prevent the occurrence of the latter.

The threat model of an adversary is depicted as in Figure 1. Assuming that $f_N(x(\mathbf{z})) = 1$, an adversary is to compute another input $\widetilde{x}(\mathbf{z})$ with a certain payoff to have a different classification, i.e., $f_N(\widetilde{x}(\mathbf{z})) = 0$. Without loss of generality, the *payoff* is measured with the norm-distance from $\widetilde{x}(\mathbf{z})$ to its original image $x(\mathbf{z})$, or formally

$$||\widetilde{x}(\mathbf{z}) - x(\mathbf{z})||_p \tag{10}$$

To deviate from an input image $x(\mathbf{z})$ to its adversarial input $\widetilde{x}(\mathbf{z})$, a large body of adversarial example generation algorithms and adversarial test case generation algorithms are available [7], [8]. Formal verification based methods such as [9]–[11] can also be used. Given a neural network N and an input x, an adversarial algorithm A produces an adversarial example $A(N, x)$ such that $f_N(A(N, x)) \neq f_N(x)$. On the other hand, for test case generation, an algorithm A produces a set of test cases $A(N, x)$, among which the optimal adversarial test case is such that $\arg\min_{\widetilde{x} \in A(N,x), f_N(\widetilde{x}) \neq f_N(x)} ||\widetilde{x} - x||_p$. We remark that, the work in this paper is independent from particular adversarial algorithms. We use in our experiments two algorithms:

- DeepFool [12], which finds an adversarial example $\widetilde{x}$ by projecting x onto the nearest decision boundary.
- DeepConcolic [13], which generates test cases by applying combined symbolic and concrete execution, guided by adapted MC/DC metrics for neural networks.

We denote by $payoff(A, N, x)$, the payoff that an algorithm A needs to compute an adversarial example from x and N. Furthermore, we assume that the adversary can observe the parameters of the Bayes filter, for example, $\mathbf{H}_k, \mathbf{F}_k$, $\mathbf{Q}_k, \mathbf{R}_k$ of the Kalman filter.

B. $\{PO\}^2$-Labelled Transition Systems

Let $Prop$ be a set of atomic propositions. A payoff and partially-ordered label transition system, or $\{\text{PO}\}^2$-LTS, is a tuple $M = (Q, q_0, kf, L, \alpha, \beta)$, where Q is a set of states, $q_0 \in Q$ is an initial state, $kf \subseteq Q \times Q$ is a transition relation, $L : Q \rightarrow 2^{Prop}$ is a labelling function, $\alpha : Q \times Q \rightarrow \mathbb{R}^+$ is a payoff function assigning every transition a non-negative real number, and $\beta : kf \rightarrow kf$ is a partial order relation between out-going transitions from the same state.

C. Reduction of WAMI Tracking to $\{PO\}^2$-LTS

We model a neural network enabled state estimation system as a $\{\text{PO}\}^2$-LTS. A brief summary of some key notations in this paper are in Table I. We let each pair $(\mathbf{s}_k, \mathbf{P}_k)$ be a state, and use the transition relation kf to model the transformation from a pair to another pair in a Bayes filter. We have the initial state q_0 by choosing a detected vehicle $(\mathbf{s}_0, \mathbf{P}_0)$ on the map. From a state $q_{k-1} = (\mathbf{s}_{k-1}, \mathbf{P}_{k-1})$ and a set Z_k of candidate observations, we have one transition (q_{k-1}, q_k) for each $\mathbf{z} \in Z_k$, where $q_k = (\mathbf{s}_k, \mathbf{P}_k)$ can be computed with Equations (3)-(5) by having $\mathbf{z}_k$ in Equation (6) as the new observation. For a state $q_k = (\mathbf{s}_k, \mathbf{P}_k)$, we write $\mathbf{s}(q_k)$ to denote the estimate $\mathbf{s}_k$, $\mathbf{P}(q_k)$ to denote the covariance matrix $\mathbf{P}_k$, and $\mathbf{z}(q_k)$ to denote the new observation that has been used to compute $\mathbf{s}(q_k)$ and $\mathbf{P}(q_k)$ from its parent state q_{k-1}.

Subsequently, for each transition (q_{k-1}, q_k), its associated payoff $\alpha(q_{k-1}, q_k)$ is denoted by $payoff(A, N, x(\mathbf{z}(q_k)))$, i.e., the payoff that the adversary uses the algorithm A to manipulate $x(\mathbf{z}(q_k))$ – the image covering the observation $\mathbf{z}(q_k)$ – into another image on which the neural network N believes there exists no vehicle.

For two transitions (q_{k-1}, q_k^1) and (q_{k-1}, q_k^2) from the same state q_{k-1}, we say that they have a partial order relation, written as $(q_{k-1}, q_k^1) \prec (q_{k-1}, q_k^2)$, if making $\mathbf{z}(q_k^2)$ the new observation requires the adversary to fool the network N into misclassifying $x(\mathbf{z}(q_k^1))$. For example, in WAMI

Notations	Description
$\mathbf{z}_k \in Z_k$	observed location by WAMI tracking
$x(\mathbf{z})$	an $d_1 \times d_2$ image covering location $\mathbf{z}$
f_N	neural network function
$payoff(A, N, x)$	payoff for algorithm A computing an adversarial example from x and N
$q_k = (\mathbf{s}_k, \mathbf{P}_k)$	a state at step k, consisting of estimate and covariance matrix
$\mathbf{s}(q_k)$, $\mathbf{P}(q_k)$ and $\mathbf{z}(q_k)$	estimate of q_k, covariance matrix of q_k and observed location for transition (q_{k-1}, q_k)
ρ	a path of consecutive states $q_l ... q_u$

TABLE I: A Summary of Notations Used

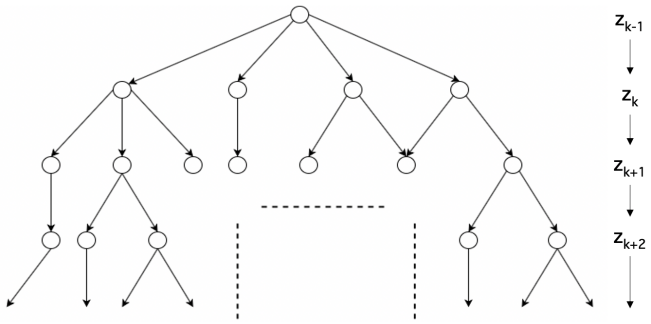


Fig. 2: Tree diagram of an unfolding $\{\text{PO}\}^2$-LTS

tracking, according to Equation (6), the condition means that $||\mathbf{z}(q_k^2) - \mathbf{z}||_p > ||\mathbf{z}(q_k^1) - \mathbf{z}||_p$, where $\mathbf{z} = \mathbf{H}_k \mathbf{s}(q_{k-1})$ is the predicted location.

Example 1. *Figure 2 depicts a tree diagram for the unfolding of a labelled transition system. The root node on top represents the initial state q_0. Each layer comprises all possible states of $q_k = (\mathbf{s}_k, \mathbf{P}_k)$ at step k of WAMI tracking, with $\mathbf{s}_k$ being one possible estimate, and $\mathbf{P}_k$ the covariance matrix. Each transition connects a state q_{k-1} at step $k-1$ to q_k at step k. $\ldots, \mathbf{z}_{k-1}, \mathbf{z}_k, \mathbf{z}_{k+1}, \mathbf{z}_{k+2}, \ldots$ are the observed locations at each step by WAMI tracking.*

Given a $\{\text{PO}\}^2$-LTS M, we define a path ρ as a sequence of consecutive states $q_l ... q_u$, and $\mathbf{z}(\rho)$ as a sequence of corresponding observed location $\mathbf{z}_l ... \mathbf{z}_u$ for $0 \leq l < u$, where l and u are the starting and ending time under consideration, respectively. We write ρ_k for the state q_k, and $\mathbf{z}(\rho_k)$ for the observed location $\mathbf{z}(q_k)$ on the path ρ.

D. A Simple Monitor on Bayesian Uncertainty

Given the convergence of the KF (as explained in Section II-D), we can easily design a system to monitor the Bayesian uncertainty: whenever there is an increase of the uncertainty range ϵ, an alarm is set to notify the potential attack. To overcome this monitor, we require that a successful attack should not present an increase on ϵ. To understand when the increase may appear for the WAMI tracking system, we recall the discussion in Section II-C that a tracking associates the nearest observation $\mathbf{z}$ within the uncertainty range ϵ at each step. When no observations are available in the range, e.g., all the observations in Z_k are attacked, only Equation (3) is performed and Equation (4) is skipped. In this case, the Bayesian uncertainty increases due to the noise $\mathbf{Q}_k$ and the transition matrix $\mathbf{F}_k$.

E. Specification as Optimization

Verification determines whether a specification ϕ holds on a given LTS M [14]. Usually, a logic language, such as CTL, LTL, or PCTL, is used to formalize the specification ϕ. In this paper, to suit our needs, we let the specification ϕ be a constrained optimisation objective, and then the verification is to determine whether, given M and ϕ, there is a solution to the optimisation problem. If the answer is affirmative, an optimal solution is returned.

We focus on the specification that expresses the robustness of the system – ***given an adversary, whether or not the state estimation system is able to function well with only minor loss of localisation precision?*** – but remark that the verification algorithm can be extended to work with other specifications.

First, we consider the measure for the loss of localisation precision. Let ρ be an original path that has not suffered an attack. We define $dist(\rho, \widetilde{\rho})$ as the distance between ρ and the other path $\widetilde{\rho}$, which is obtained after an attack, and say that the system is robust to the attack if $dist(\rho, \widetilde{\rho}) < \theta_{robustness}$ for a threshold $\theta_{robustness} > 0$. For the WAMI tracking system, we define

$$dist(\rho, \widetilde{\rho}) = (\sum_{k=l}^{u} ||\mathbf{z}(\rho_k) - \mathbf{z}(\widetilde{\rho}_k)||_p)/(u - l + 1) \quad (11)$$

as the averaged norm distance between two given times l and u. It is straightforward to see that, if we can find the maximally allowed distance $\max_{\widetilde{\rho}} dist(\rho, \widetilde{\rho})$ then we can firmly conclude – through verification – whether or not the system is robust on the path ρ, by comparing a given distance d with $\max_{\widetilde{\rho}} dist(\rho, \widetilde{\rho})$.

In addition to the satisfaction of the objective as above, we require the *best* attack to not be easily detected by e.g., monitors. In this paper, we consider two monitors who look after two quantities, as explained below. Firstly, we consider a monitor Γ which looks after the Kalman filter's state by considering its convergence. Γ has the following definition:

$$\Gamma(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k) = \begin{cases} 1 & \epsilon(\widetilde{\rho}_k) \leq \epsilon(\widetilde{\rho}_{k-1}) \\ 0 & \epsilon(\widetilde{\rho}_k) > \epsilon(\widetilde{\rho}_{k-1}) \end{cases} \quad (12)$$

Basically, Γ continuously checks the value of uncertainty ϵ, which is derived from covariance $\mathbf{P}$. It is to capture the case where the uncertainty increases, as required by Section III-D.

We consider the other simple monitor which looks after the (mean) payoff in attacking the perception system and, once the payoff is over a threshold θ_{payoff} the attacker is detected. Let $(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)$ be a transition on an attack path $\widetilde{\rho}$, we have

$$\varphi(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k) = \sum_{(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k^\diamond) \prec (\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)} \alpha(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k^\diamond) \quad (13)$$

as the *combined payoffs* that are required to implement the transition $(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)$. Intuitively, all the payoffs of the transitions $(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k^\diamond)$, which are partially ordered by the envisaged

transition $(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)$, are counted. In the WAMI tracking system, this means that the attack results in misclassifications of all the images $x(\mathbf{z}(\widetilde{\rho}_k^\circ))$ with $\mathbf{z}(\widetilde{\rho}_k^\circ)$ being closer to the predicted location $\mathbf{F}_k\mathbf{s}(\widetilde{q}_{k-1})$ than $\mathbf{z}(\widetilde{\rho}_k)$.

Therefore, the optimisation problem can be formulated as

$$\begin{aligned} \underset{\widetilde{\rho}}{\textbf{maximize}} \quad & dist(\widetilde{\rho}, \rho) \\ \textbf{subject to} \quad & \sum_{k=l+1}^{u} \Gamma(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k) = u - l \\ & \sum_{k=l+1}^{u} \varphi(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k) \leq (u-l) \cdot \theta_{payoff} \end{aligned} \tag{14}$$

where $dist(\widetilde{\rho}, \rho)$ is the deviation from the original track ρ to the adversarial track $\widetilde{\rho}$, $\sum_{k=l+1}^{u} \Gamma(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)$ is the monitoring of convergence of tracking to enable $\widetilde{\rho}$, and

$$\varepsilon_{avg} = \sum_{k=l+1}^{u} \varphi(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)/(u-l) \tag{15}$$

is the mean payoff from time l to u. Finally, the verification problem is to compute the above optimisation objective on a given $\{\text{PO}\}^2$-LTS M.

IV. Automated Verification

An attack on the WAMI system, as in Section III-A, adds perturbations to the images containing vehicles in order to fool the neural network into making a wrong detection. Then, this wrong detection will be passed on to the KF-based tracking system. For the KF, a detection is adopted as an observation w.r.t. the Gnn, as explained in Section II-C. In this section, we develop algorithms to find the maximally deviated path, which will be compared with a given threshold $\theta_{robustness}$ to have the verification result.

A. Baseline Method

We consider a baseline method that does not take into account the monitor on Bayesian uncertainty (Equation (11)) or the other monitor on attack payoff (Equation (13)); it simply attacks the neural networks to make the currently associated observation – the nearest one to the prediction – unseen to the KF. This is the method used in [5].

B. Verification Algorithm based on Exhaustive Search

Our verification algorithm proceeds by exhaustively computing over all possible attacked tracks. Since a final deviation is not available until the end of a simulation, the tree has to be fully expanded from the root to the leaf and all the paths are explored. Breadth-first search (BFS) is used to find the best solution.

The details are in Algorithm 1. We have several operation functions on the tree diagram for the labelled transition system. $leaf$ returns all leaf nodes of the given root node. $parent$ associates a node to its parent node. $path$ returns all tree paths from the given root node to the leaf nodes.

Lines 2-15 present the procedure of constructing the tree diagram. First, we set the root node ρ_l (Line 2), that is, we will attack the system from the (l)-th state of the original path ρ and enumerate all possible adversarial tracks. At each step k, function $neighbours$ will list all observations near the predicted location (Line 5). Then, each observation is incorporated with current state ρ_k for the calculation of next state ρ_{k+1} (Line 7). To enable each transition (ρ_k, ρ_{k+1}), the partial order relation is followed when attacking the system and recording the payoff φ; also, the convergence property of KF is checked (Line 8-11). If these constraints are satisfied, the potential ρ_{k+1} is accepted and added as the child node of ρ_k. Once the tree is constructed, we continue simulating the tracks to the end of time, $k = n$, (Lines 16-17). Finally, all the paths are compared with the original one to select the most deviated path, satisfying the payoff constraint (Lines 18-19).

Algorithm 1: Verification Algorithm based on BFS

Input: LTS model M, n, l, u
Output: The most deviated path $\widetilde{\rho}^M$

1: calculate the original path ρ from $k = 0$ to $k = n$
2: set ρ_l as root node
3: **for** k from l to $u-1$ **do**
4: **for** each node $\widetilde{\rho}_k$ in $leaf(\rho_l)$ **do**
5: find potential observations $Z \leftarrow neighbours(\widetilde{\rho}_k)$
6: **for** each observed location $\mathbf{z}$ in Z **do**
7: $\widetilde{\rho}_{k+1} \leftarrow kf(\widetilde{\rho}_k, \mathbf{z})$
8: calculate the attack payoff $\varphi(\widetilde{\rho}_k, \widetilde{\rho}_{k+1})$
9: **if** $\Gamma(\widetilde{\rho}_k, \widetilde{\rho}_{k+1}) \neq 1$ **then**
10: break
11: **end if**
12: $\widetilde{\rho}_k = parent(\widetilde{\rho}_{k+1})$
13: **end for**
14: **end for**
15: **end for**
16: $P \leftarrow path(\rho_l)$
17: calculate the path $\widetilde{\rho}$ in set P to $k = n$
18: $\varepsilon = \sum_{k=l+1}^{u} \varphi(\widetilde{\rho}_{k-1}, \widetilde{\rho}_k)/(u-l)$
19: $\widetilde{\rho}^M \leftarrow \arg\max_{\widetilde{\rho} \in P, \varepsilon(\widetilde{\rho}) \leq \theta_{payoff}} dist(\rho, \widetilde{\rho})$
20: **return** $\widetilde{\rho}^M$

C. Heuristic Algorithm based on Sub-optimal Greedy Search

Although the verification algorithm can find the optimal solution, its computation is not polynomial time, and therefore cannot be executed in real-time. A heuristic function can be designed to rank all possible children nodes to select the most likely one. As shown in Algorithm 2, heuristic search is based on the strategy of making the locally optimal choice at each stage. Compared with the exhaustive exploration, heuristic search can find the sub-optimal solution with significantly less computational cost.

To design a heuristic search algorithm for the optimization problem (14), we construct two KFs which run simultaneously for the vehicle tracking. One normal KF accepts the original observations and outputs its states to guide the selection of observations by the adversarial KF. The locally optimal choice is to select the most distant observation of

Algorithm 2: Greedy Based Heuristic Search

Input: LTS model M, n, l, u
Output: The deviated path $\widetilde{\rho}$

1: calculate the original path ρ from $k=0$ to $k=n$
2: start from $\widetilde{\rho}_l = \rho_l$
3: **for** k from l to $u-1$ **do**
4: find potential observations $Z \leftarrow neighbours(\widetilde{\rho}_k)$
5: $\mathbf{z}(\widetilde{\rho}_k) \leftarrow g(Z, \mathbf{z}(\rho_k))$
6: $\widetilde{\rho}_{k+1} \leftarrow kf(\widetilde{\rho}_k, \mathbf{z}(\widetilde{\rho}_k))$
7: calculate the payoff $\varepsilon \leftarrow \varphi(\widetilde{\rho}_k, \widetilde{\rho}_{k+1})$
8: **if** $\Gamma(\widetilde{\rho}_k, \widetilde{\rho}_{k+1}) \neq 1$ or $\varepsilon > \theta_{payoff}$ **then**
9: remove $\mathbf{z}(\widetilde{\rho}_k)$ from Z
10: jump to Line 5
11: **end if**
12: **end for**
13: calculate the path $\widetilde{\rho}$ to $k=n$
14: **return** $\widetilde{\rho}$

the original one (Line 5). The heuristic function is

$$g(Z, \mathbf{z}) = \arg\max_{\widetilde{\mathbf{z}} \in Z} ||\widetilde{\mathbf{z}} - \mathbf{z}||_2 \tag{16}$$

where $\mathbf{z}$ and $\widetilde{\mathbf{z}}$ are the correct and adversarial observation of tracked vehicle obtained in detection, respectively. Lines 7-11 monitor the Bayesian uncertainty and the attack payoff. If the current solution cannot bypass the monitor, the heuristic algorithm will iterate to find the next most distant observation until a feasible one.

V. Experimental Results

We conduct a set of experiments to show the effectiveness of our verification algorithm (Algorithm 1) and heuristic search algorithm (Algorithm 2) in the WAMI tracking system. We believe our approaches can be generalised to work with other systems using both Bayes filter(s) and neural networks.

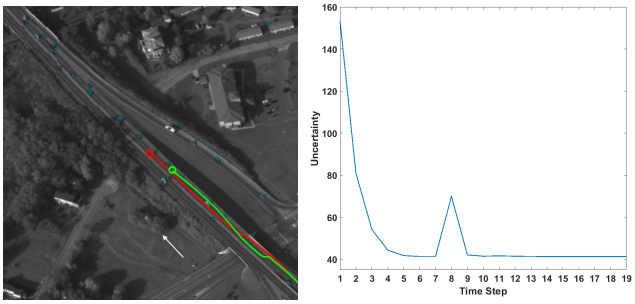
A. Research Questions

Our evaluation experiments are guided by the following three research questions.

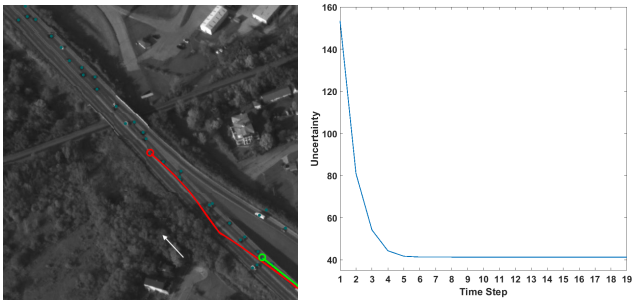
RQ1 Does the awareness to the Bayesian uncertainty and the input perturbation in our algorithms improve the quality of the obtained solutions?
RQ2 Can our algorithms prove the robustness of the system?
RQ3 What are the pros and cons of the two algorithms?

B. Experimental Setup

We consider a number of original tracks with maximum length of 20 steps ($n = 19, k \in [0, 19]$). An attack on the system is conducted between time steps l and u, denoted as $Attack(l, u)$. The original track is colored in *green* in both the high-resolution images (Figure 3–4) and the state space unfolding (Figure 5). The attacked track is colored in *red*. The white-color arrows in the high-resolution images indicates the ground-truth direction of the vehicle.



(a) Baseline, $\varepsilon_{avg} = 1.231$, $T = 70$, $dist = 42$



(b) Heuristic/verification, $\varepsilon_{avg} = 0.676$, $T = 73$, $dist = 189$

Fig. 3: The comparison between the baseline and the heuristic/verification in selected scene with configuration $Attack(5, 8)$

Moreover, in all experiments, we record the following measures for each attack: mean payoff, ε_{avg}, mean deviation, $dist$, as defined in Section III-E, and runtime T (seconds). The payoff threshold is set at $\theta_{payoff} = 1$ and the robustness threshold is set at $\theta_{robustness} = 100$. The thresholds are hyper-parameters and can also be user-defined. Here, we run 100 test scenes and set the mean values for the thresholds.

C. Returning Good Solutions within Constraints (RQ1)

In Section III-E, we set the the criteria for the adversary to make the attack more realistic. From the adversary's perspective, constraints make sure the attack is not easily detected by simple monitors. In this section, we discuss the impact of these constraints by comparing the two algorithms with the baseline method.

Figure 3 presents two plots displaying the change of uncertainty ϵ over time, for baseline method and heuristic/verification algorithm, respectively. In this test scene, heuristic and verification algorithms output the same results. Since the baseline method does not take into consideration the KF's convergence, there exists the situation that, in some time step, no observations are available within the search range – because the only possible observation is attacked – and there is a significant fluctuation at step 8 in the plot.

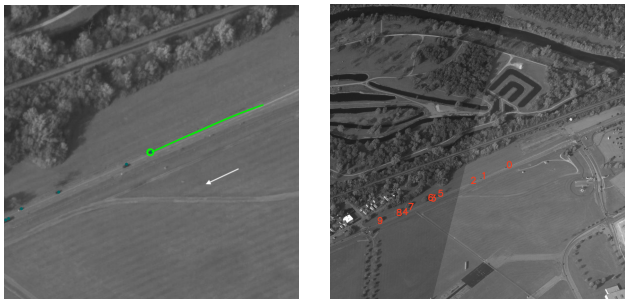
For the impact of input perturbations, we can see that the heuristic/verification algorithm has a much smaller ε_{avg} – representing a lower mean perturbation cost – than the baseline method. The runtime T is related to the attacking strategy. The heuristic/verification algorithm may need to attack multiple images at each time step to make sure that the

remotest observation is taken as the observation. Therefore, the generation of perturbations to attack neural networks may take slightly more time than the baseline method. Overall, the heuristic/verification algorithm can find better attacking solution with smaller distance to the original track than the baseline method, and at the same time it satisfies the constraints from WAMI monitors.

D. Proof of Robustness Against the Attack (RQ2)

In addition to the ability of finding counterexamples to the robustness property, one may be interested in whether or not our approach can prove, with guarantee, that a system is robust. Figure 4 provides an example. We choose the test scene in Figure 4b, where the red numbers represent the detected moving vehicles. We apply heuristic search and verification algorithms on the track of vehicle No.0, which is shown in Figure 4a. In this case, the WAMI system shows the robustness against the attack. Fundamentally, there are few other vehicles around the tracking car and, at each attack step, only one observation is available in the search range, shown in Figure 4c. Thus, to build the tree diagram for this problem, there is only one path to be traversed and therefore no adversarial path is possible.

While the above seems to be an extreme case, it actually represents a typical class of systems that tend to be more robust than others, i.e., systems for the test scenes with few external disturbances. The external disturbance comes from the observations of the surrounding vehicles, providing the wrong measurements when KF updates. We remark that this proof can be generalised to exercise the system's robustness for more complicated environments, consisting of an extensive number of vehicles.



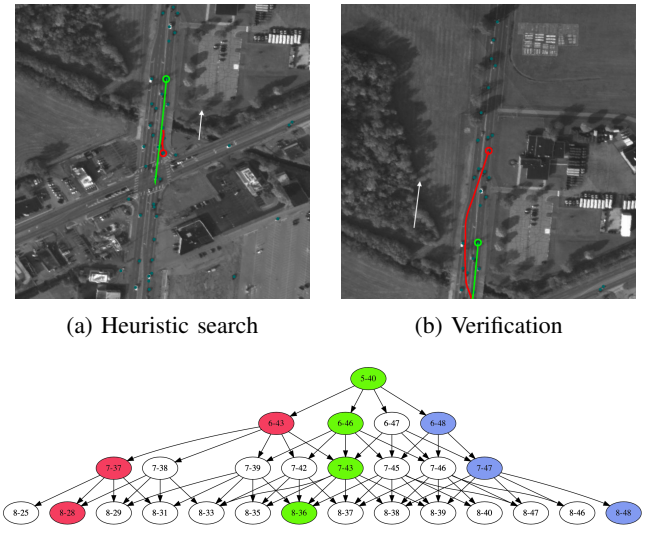
(a) Output (b) Detected moving vehicles

(c) Transition of path for track in (a)

Fig. 4: Failure in finding an attacked path

E. Pros and Cons of the Two Algorithms (RQ3)

We compare the performance of the two algorithms – verification and heuristic – by choosing the same tracking start point and apply the algorithms to the same time interval. The detailed running results are presented in Figure 5 and Table II. As depicted in Figure 5c, the verification method enumerates all possible tracks and selects the most deviated one, colored red. This is the optimal solution to the optimization problem. While the heuristic search method can find a sub-optimal track, colored blue.



(a) Heuristic search (b) Verification

(c) Enumeration of all possible Tracks

Fig. 5: Heuristic search and verification $Attack(5, 8)$ on a selected scene. Tree graph exhibits all possible tracks, where green is the original track, blue is the attacked track found by heuristic search, and red is the attacked track found by verification. The labels on the nodes represent "(time step)-(ID of associated detection)".

TABLE II: Tracks Generated by Different Algorithms.

Original Track	t	5	6	7	8		19
	ID	40	46	43	36	...	11
	$dist$	0	0	0	0		0
Adv. Track under heuristic search	t	5	6	7	8		19
	ID	40	48	47	48	...	58
	$dist$	0	3.4	10.1	19.7		170.3
Adv. Track under verification	t	5	6	7	8		19
	ID	40	43	37	28	...	7
	$dist$	0	0.4	5.1	13.7		193.3

The data in Table II more precisely indicates the transition policy of the two algorithms. During the time interval $(5, 8)$, the heuristic search guides the tracking to the locally optimal waypoint, which has larger $dist$ values than the verification method. However, for the long term (over 20 time steps), verification is always able to determine the optimal solution.

We also evaluate the two algorithms statistically by sampling 100 test scenes and calculate the average performance. The results can be found in Table III. Since we incorporate the same adversarial attack algorithm, both verification and heuristic search algorithms have similar average perturbation cost ε_{avg}. However, for the runtime cost T, verification is significantly more time-consuming than the heuristic search.

TABLE III: Statistical Comparison between the Verification and Heuristic Search Algorithms

Algorithm	ε_{avg}	T	$dist$	Probability of Finding Best Adv. Track
heuristic search	0.63	78	93	80%
verification	0.65	3465	117	100%

Most test scenes have a high concentration of vehicles, i.e., there are large amount of candidate observations. The runtime cost of verification is proportional to the candidate observations. Therefore, while verification can find the complete results, it is not suitable for real-time analysis. In contrast, heuristic search, although unable to guarantee the optimal solution, is efficient in runtime. Actually, in our experiments, for 80% of the cases we studied, the heuristic search algorithm can find the optimal solution.

In terms of the safety risks of the WAMI tracking system, we noticed that, the potential risk from the learning components is non-trivial. For example, we often see e.g., a 3-step attack lead to a significantly deviated tracking – a deviation distance of 117 from the original track on average. This illustrates the lack of robustness within the state estimation system we have verified.

VI. Related Work

We reviewed research on the safety analysis of learning enabled autonomous systems. Indeed, ML techniques have been confirmed to have potential safety risks [8]. Currently, most safety analysis work is on the verification and validation of ML components. Please refer to [15] for a recent survey on the progress of this area.

Research is sparse at the system level, and none on state estimation systems. In [16], a compositional framework is developed for the falsification of temporal logic properties of cyber physical systems with ML components. Alternatively, a simulation based approach [17] is suggested to verify the barrier certificates – representing safety invariants – of autonomous driving systems with an SMT solver. In both papers, the interaction – or synchronisation – between ML and other components is through a shared value, which is drastically different with neural network enabled state estimation, where the synchronisation is closer to the message-passing regime. Moreover, the erroneous behaviours and the specifications of this paper are different from those of [16], [17]. These differences suggest that the existing approaches cannot be extended to work with our problem.

Moreover, there is some early research on the robustness of KF by false information injection [18], where the false information is modelled as Gaussian noise, differing from our consideration of adversarial attacks.

VII. Conclusion

In this paper, we propose a practical verification approach for an in-depth investigation into the safety risks of a typical class of learning-enabled systems in robotics. Extensive experiments are conducted on a real-world system – WAMI tracking – which tracks ground vehicles through high-resolution imagery inputs. The results show that our approach can prove the safety of the system against attacks to the neural network components; if it is safe; and returns the best counterexample otherwise.

References

[1] G. Papadopoulos, M. F. Fallon, J. J. Leonard, and N. M. Patrikalakis, "Cooperative localization of marine vehicles using nonlinear state estimation," in *IROS*, pp. 4874–4879, IEEE, 2010.

[2] N. Gordon, D. Salmond, and C. Ewing, "Bayesian state estimation for tracking and guidance using the bootstrap filter," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 6, pp. 1434–1443, 1995.

[3] Y. Zhou and S. Maskell, "Detecting and tracking small moving objects in wide area motion imagery (wami) using convolutional neural networks (cnns)," in *2019 22th International Conference on Information Fusion (FUSION)*, pp. 1–8, July 2019.

[4] W. He and Y. Dong, "Adaptive fuzzy neural network control for a constrained robot using impedance learning," *IEEE transactions on neural networks and learning systems*, 2017.

[5] Y. Sun, Y. Zhou, S. Maskell, J. Sharp, and X. Huang, "Reliability validation of learning enabled vehicle tracking," in *ICRA*, 2020.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[7] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Structural test coverage criteria for deep neural networks," *ACM Transactions on Embedded Computing Systems*, 2019.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[9] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *CAV*, 2017.

[10] W. Ruan, X. Huang, and M. Kwiatkowska, "Reachability analysis of deep neural networks with provable guarantees," in *IJCAI*, pp. 2651–2659, 2018.

[11] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, "Global robustness evaluation of deep neural networks with provable guarantees for hamming distance," in *IJCAI*, 2019.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

[13] Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, and D. Kroening, "Concolic testing for deep neural networks," in *ASE*, 2018.

[14] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model checking*. The MIT Press, 2018.

[15] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, 2020.

[16] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Systematic testing of convolutional neural networks for autonomous driving," *arXiv preprint arXiv:1708.03309*, 2017.

[17] C. E. Tuncali, J. Kapinski, H. Ito, and J. V. Deshmukh, "Reasoning about safety of learning-enabled components in autonomous cyber-physical systems," *arXiv preprint arXiv:1804.03973*, 2018.

[18] R. Niu and L. Huie, "System state estimation in the presence of false information injection," in *Statistical Signal Processing Workshop (SSP)*, pp. 385–388, IEEE, 2012.